Javascript avancé
Module 5 – Angular (init.)

OBJECTIFS

Comprendre les bases d'Angular

Utiliser des services avec Angular

Contacter une API

SOMMAIRE

Présentation

Historique

Nom tiré des angular brackets <>

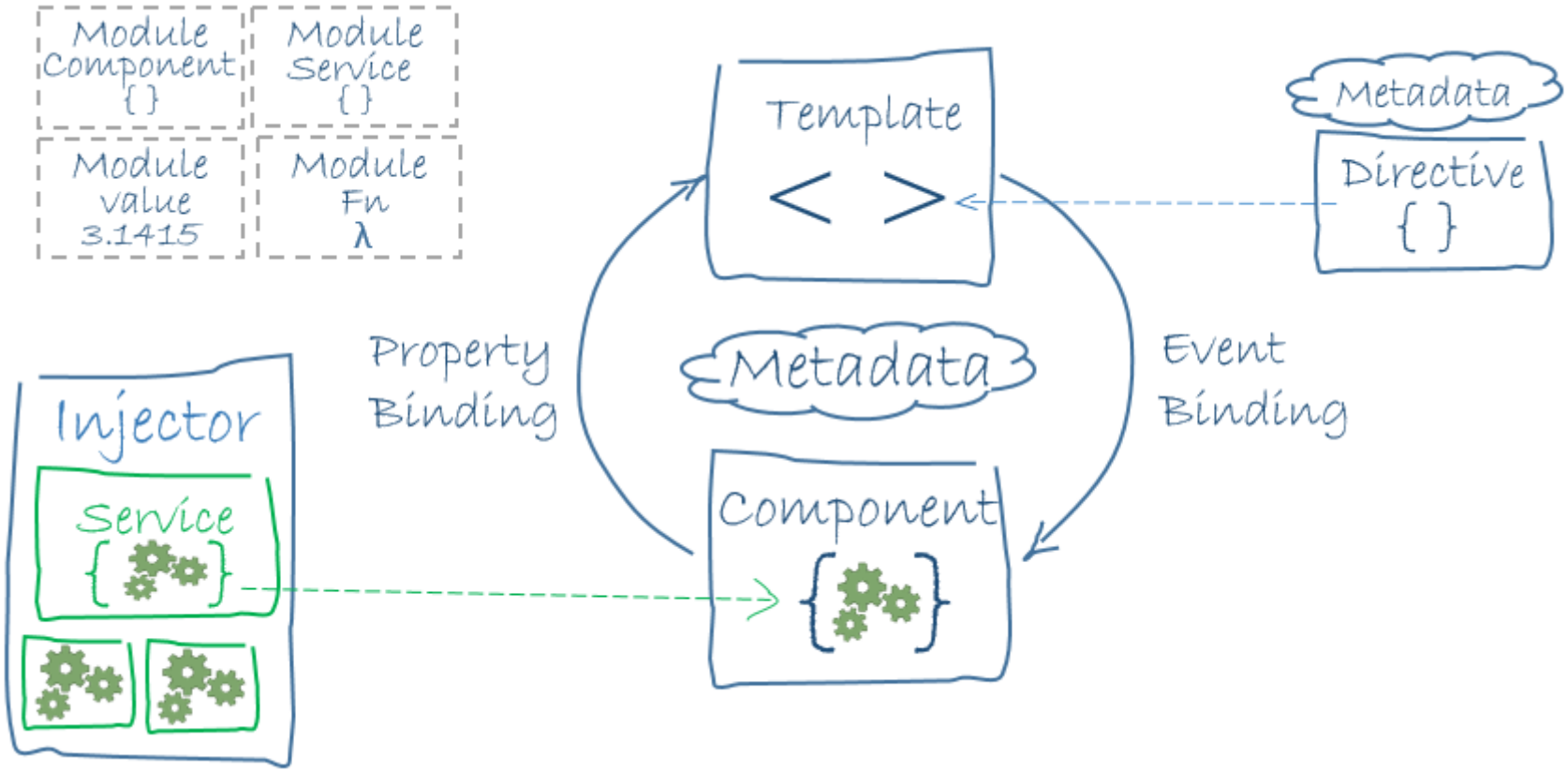
En 2010 : AngularJs basé sur Javascript

Depuis 2016 : Angular v2+ basé sur Typescript

Réécriture et refonte totale

https://angular.io

Architecture



Command Line Interface

Angular CLI

Utilitaire en ligne de commande

Permet de :

- Créer un projet
- Tester un projet
- Lancer un serveur web de développement
- Construire la version en production

https://github.com/angular/angular-cli

Commandes du CLI

Installation

```
npm install -g @angular/cli
```

Création d'un projet

```
ng new demonstration-ng
```

Visualisation d'un projet

```
cd demonstration-ng
ng serve
```

Maquette

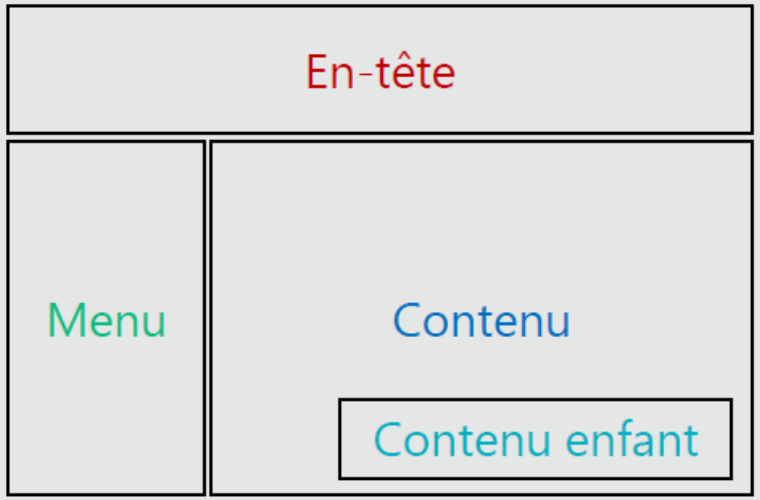
Maquette

Combien de parties différentes sur cette maquette ?

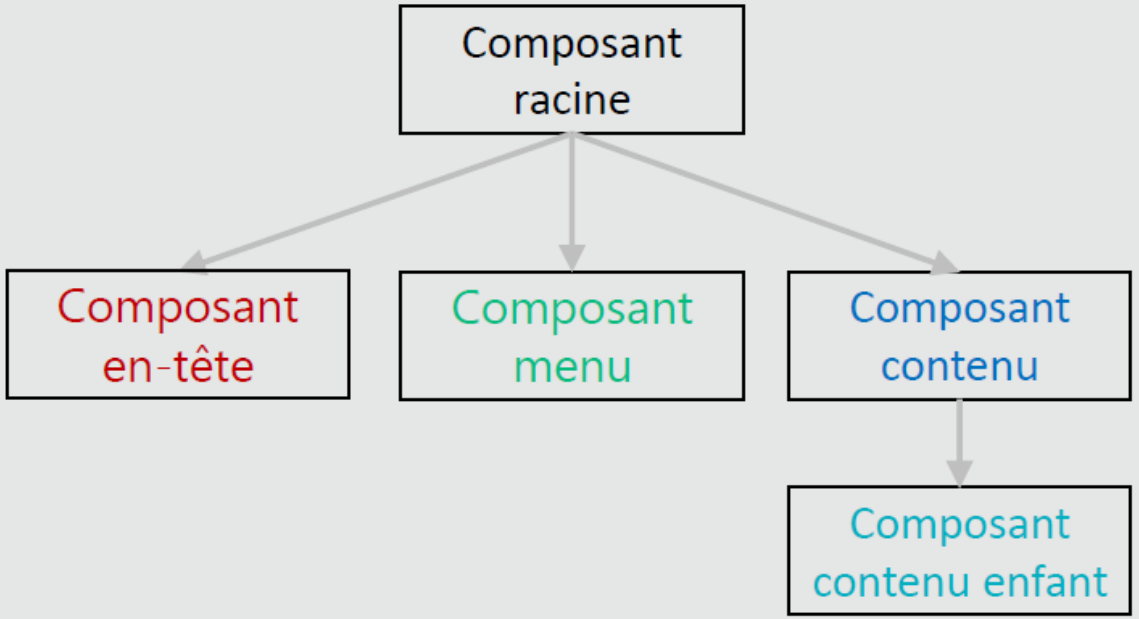
1, 2, 3, 4, 5, 6, 7 ?

Maquette

- Représentation graphique



- Représentation par composants



Composants

Créer un composant

Un composant est un dossier contenant :

- Un fichier Typescript
- Un fichier de style
- Un fichier HTML
- Un fichier de tests unitaires

```
ng generate component composants/mon-composant
```

Typescript

```typescript
import { Component } from '@angular/core';

@Component({
  selector: 'app-mon-composant',
  templateUrl: './mon-composant.component.html',
  styleUrls: ['./mon-composant.component.scss']
})

export class MonComposantComponent {
}
```

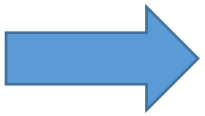
HTML

```
<p>
  Le monde se divise en deux catégories:
  ceux qui ont un pistolet chargé et ceux qui creusent.

  Toi tu creuses.
</p>
```

Liaisons entre composants et HTML

Composant => template

```ts
import {Component} from '@angular/core';

@Component({
  selector: 'app-interpolation',
  templateUrl: './interpolation.component.html',
  styleUrls: ['./interpolation.component.scss']
})
export class InterpolationComponent {
  public titre: string = 'Unsplash';
  public image: string = 'https://source.unsplash.com/random';
}
```



```html
<p>{{ titre }}</p>

<img src="{{ image }}" alt="unsplash_random"> <!-- bien -->
<img [src]="image" alt="unsplash_random"> <!-- meilleur -->
```

Template => composant

```html
<p>{{ compteur }}</p>
<div>
  <input type="submit" value="+" (click)="incremente()">
</div>
```

```typescript
import {Component} from '@angular/core';

@Component({
  selector: 'app-evenement',
  templateUrl: './evenement.component.html',
  styleUrls: ['./evenement.component.scss']
})
export class EvenementComponent {

  public compteur: number;

  constructor() {
    this.compteur = 0;
  }

  public incremente(): void {
    this.compteur++;
  }
}
```

Paramètre d'entrée

```
import {Component, Input} from '@angular/core';

@Component({
  selector: 'app-input',
  templateUrl: './input.component.html',
  styleUrls: ['./input.component.scss']
})

export class InputComponent {

  @Input() public prenom: string;

  constructor() {
    this.prenom = 'Bob Morane'; // Valeur par défaut
  }
}
```

```
<app-input prenom="Julie"></app-input>
<app-input prenom="Virginie"></app-input>
```

Directives

*ngIf et *ngFor

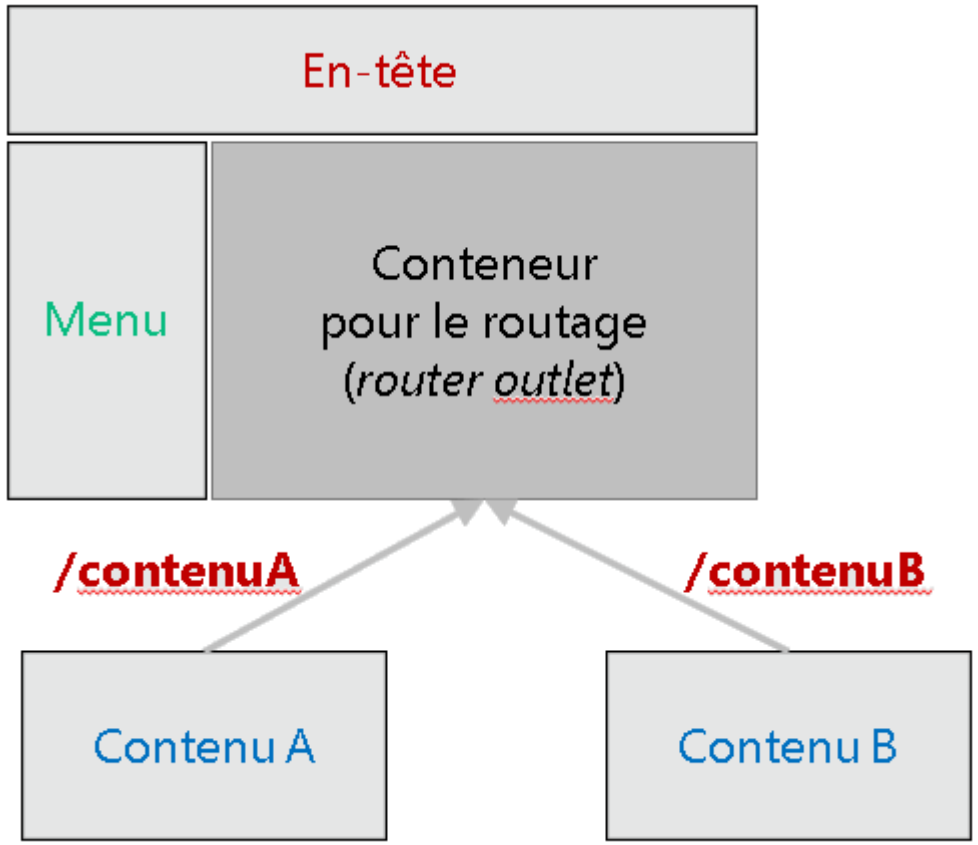
Les directives modifient la structure d'un document

- *ngIf
- *ngFor

```
<p *ngIf="<instruction typescript>">
  Affichage conditonné de la balise p
</p>

<p *ngFor="let t of tableau">
  Répétition de la balise p autant de fois que d'éléments dans le tableau
</p
```

Routage

Routage




```
// app.module.ts
import {RouterModule, Routes} from '@angular/router';

const appRoutes: Routes = [
    {path: 'contenuA', component: ContenuAComponent},
    {path: '**', component: InconnuComponent}
];

@NgModule({
imports: [
    RouterModule.forRoot(appRoutes)
    ]
  })
export class AppModule { }
```

```
<body>
  <a routerLink="/contenuA">Contenu A</a>
  <a routerLink="/contenuB">Contenu B</a>
  <div>
  <router-outlet></router-outlet>
  </div>
</body>
```

DÉMONSTRATION

Composants Angular

TRAVAUX PRATIQUES

Pokemon Partie 1 & 2

Services

Les services

Les services gèrent les traitements métier

- Récupération d'informations sur le serveur
- Authentification
- Validation
- Log
- …

L'objectif est d'augmenter la modularité de l'application Angular.

Services

```
@Injectable({
  providedIn: 'root'
})
export class MonService {
}
```

```
export class MonComposant {

  constructor(
    @Inject(MonService) private svc: MonService
  ) {
  }
}
```

DÉMONSTRATION

Services

API REST

API REST

- Interroger un serveur web
- HttpClient

```
import {HttpClient} from '@angular/common/http';

export class CrudHttpService {
    constructor(private http: HttpClient) {
    }
}
```

- Traitement asynchrone

```
Observable<any>
```

Api

C'est grâce à HTTPClient qu'une requête est exécutée :

```
export class ApiService {

  private URL: string = 'https://mon_api.com/';

  constructor(
    @Inject(HttpClient) private http: HttpClient,
  ) {
  }

  public getApi(): Observable<string> {
    return this.http.get<string>(this.URL);
  }
}
```

Observable et HTML

```ts
import {Component, Inject} from '@angular/core';
import {ApiService} from "../../services/api.service";
import {Observable} from "rxjs";

@Component({
  selector: 'app-cours',
  templateUrl: './cours.component.html',
  styleUrls: ['./cours.component.scss']
})
export class CoursComponent {

  public reponseApi$: Observable<string>;

  constructor(
    @Inject(ApiService) private api: ApiService
  ) {
    this.reponseApi$ = this.api.getApi();
  }
}
```

```html
<div *ngIf="reponseApi$ | async as reponseApi">
  <div>{{ reponseApi }}</div>
</div>
```

DÉMONSTRATION

HttpClient

TRAVAUX PRATIQUES

Pokemon Partie 3 & 4

Conclusion

- Vous connaissez les bases d'Angular
- Vous savez utiliser des services
- Vous savez contacter une API et traiter les données